CSCD01 Deliverable 1

Team 34: 3+3

The subproject within Mozilla that we have decided to work on is donate-wagtail, a content management system and web application that handles Mozilla's donations. It is a customized fork of Wagtail CMS, which is built on the Django web framework.

The production version of the donation frontend can be found at https://donate.mozilla.org/. The production-level Wagtail CMS is only accessible by Mozilla employees and volunteers, but an instance with mock data can be run by developers locally.

Architecture of the System

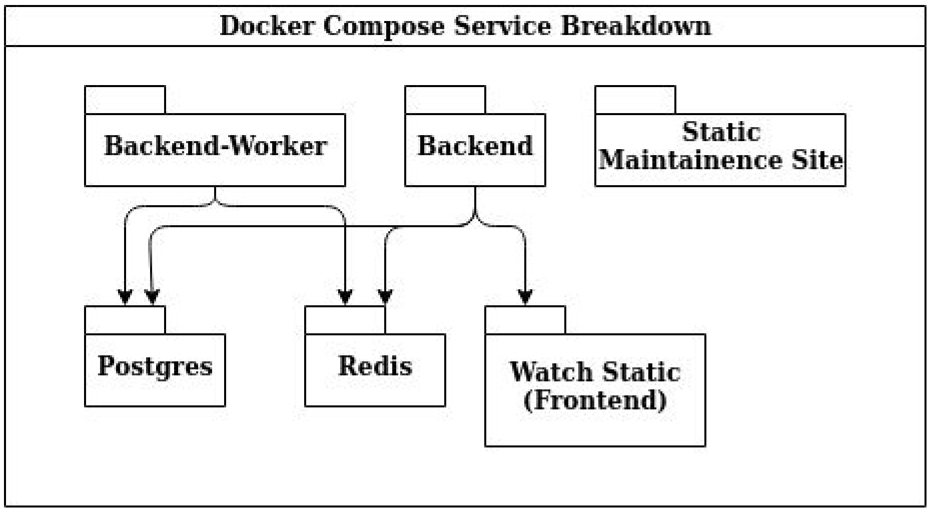


Figure 1: Docker Compose Service Breakdown.

The development and deployment environments are run with Docker containers and Docker Compose. Here is a UML diagram showing a very high overview of the different services the application runs and their dependencies, taken from the Docker Compose files.

- Backend and backend-worker are Docker services required for the Django application to run.
- Postgres and Redis are run as Docker services pulled from the main officially hosted images.
- The React frontend is built from files in the source/ folder, which are the files that the Watch Static service watches changes for.
- The Static Maintenance site is a static HTML website that is displayed whenever the website is down for maintenance.

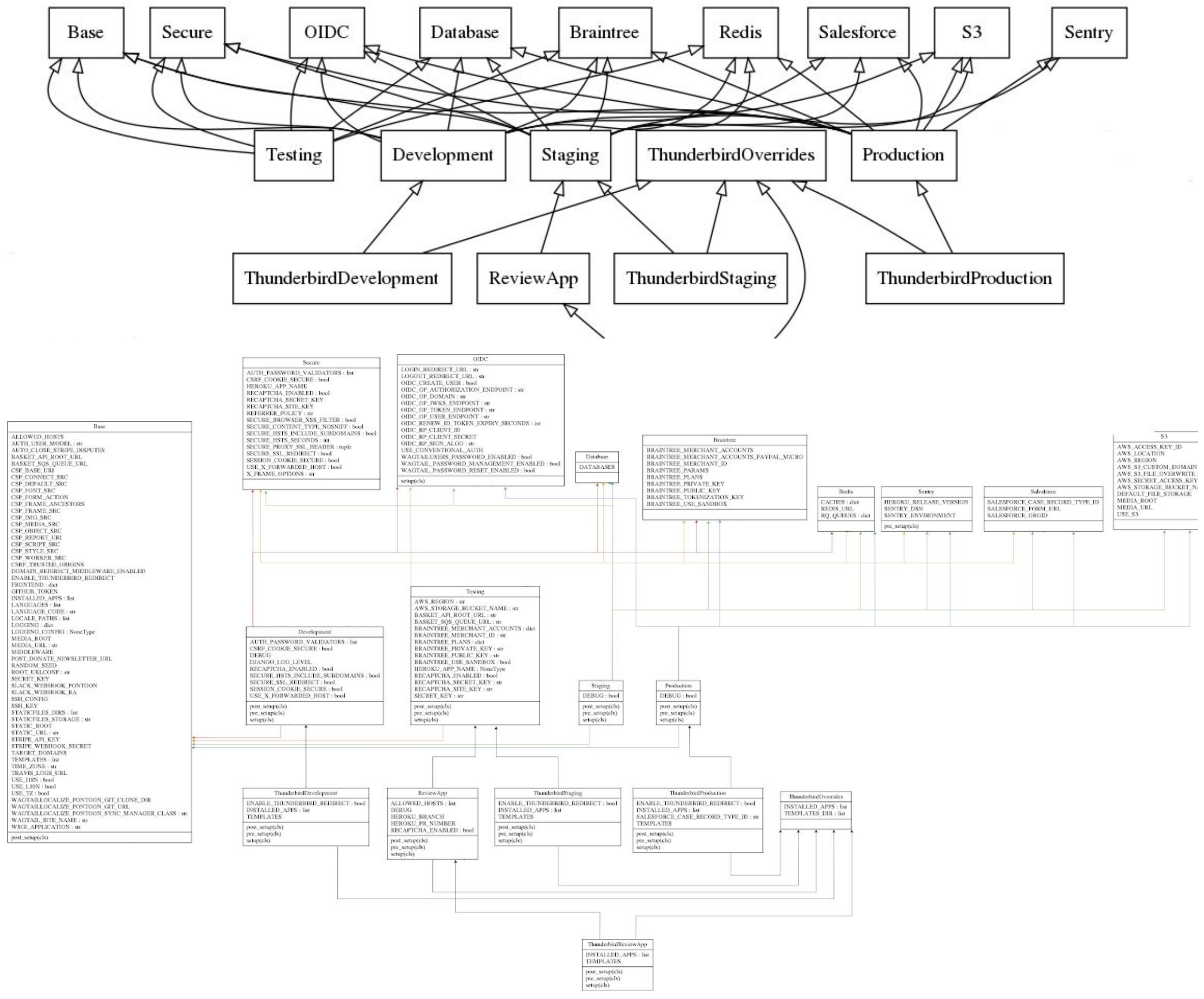


Figure 2: Backend Django configurations and dependencies.

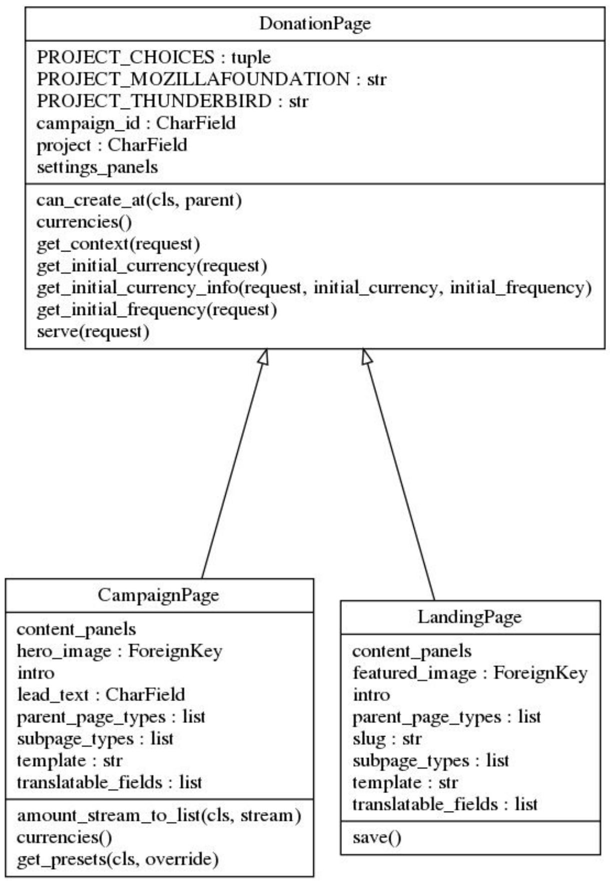
Here is a UML diagram for the backend and its dependencies, in the Django application. The dependency lines are not that important in this diagram, as all environments depend on the top-level dependencies.

In the first line, Base and Secure are simply Django's environment variables, but the rest of the classes are the application's dependencies:

- OIDC is OpenID Connect, which is how donate-wagtail handles authentication.
- Braintree is an API used for handling payments from donors.
- Redis is an in-memory cache.
- Salesforce is a CRM platform for analytics on donation trends.
- Database is a Postgres database for storing donator's information and some page information.
- S3 and Sentry are for deployment in production.

In the second and third lines are the different Django configurations for the project. The typical Testing, Development, Staging and Production environments are present, but interestingly there are also Thunderbird-specific configurations. It turns out that donate-wagtail is not only the basis for https://donate.mozilla.org/, but also https://give.thunderbird.net/en-US/! It is interesting that Mozilla has a separate donation website for Thunderbird, but not Firefox.

Figure 3: Frontend donation pages.



This is the architecture for the main pages of the donation website, including the front landing page (see here).

Donations are allowed to be in multiple different currencies, shown here in the currencies() methods, hard-coded in the back-end.

One problem is that the actual front-end of the web page (like the intro text, hero_image, lead_text in CampaignPage) is coupled with the backend. Ideally the model should be decoupled from the view.

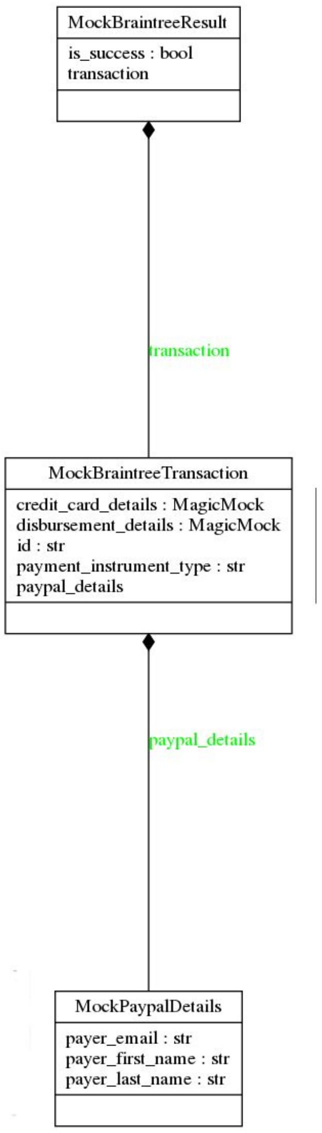


Figure 4: Mock data classes for testing payment transactions.

Here is a small example of how testing works in the donate-wagtail project. The project relies on the unittest python module, and mock data is used to simulate user and payment details instead of using real data. Shown here is the mock data used to simulate a Paypal/Braintree transaction. This is good practice by using mock data, especially with payment information.

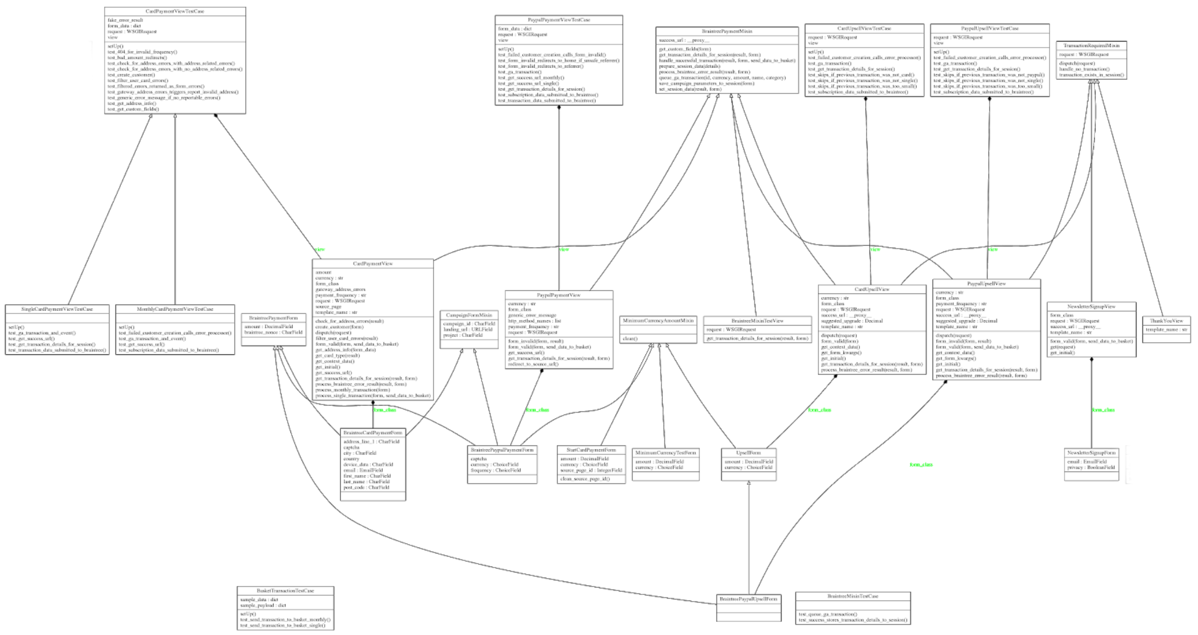


Figure 5: Payment system overview.

Different views are used for different payment methods. There are a few key views: like PaypalPaymentView and CardPaymentView. To run securely, donate-wagtail outsources the actual transactions to Braintree using the BraintreePaymenMixin. Different test classes are also coupled with the payment views. Note that under the BraintreePaypalPaymentForm, captchas are used for security purposes.

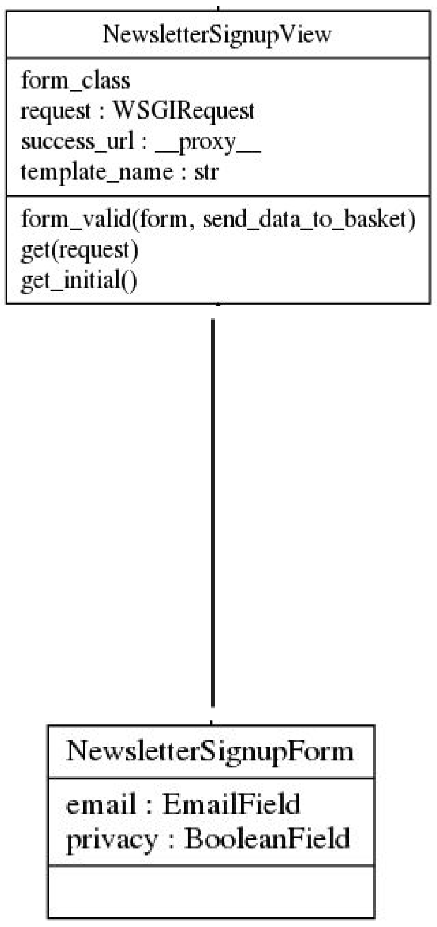


Figure 6: Newsletter form.

Donators can also choose to donate using a newsletter sign up form. The request is sent in through the Django backend, which parses the email and adds it to the subscriber database.

Suggestions for Architecture Improvement

- The big suggestion is that they migrate their frontend over from react to django or vice versa so it is a lot cleaner

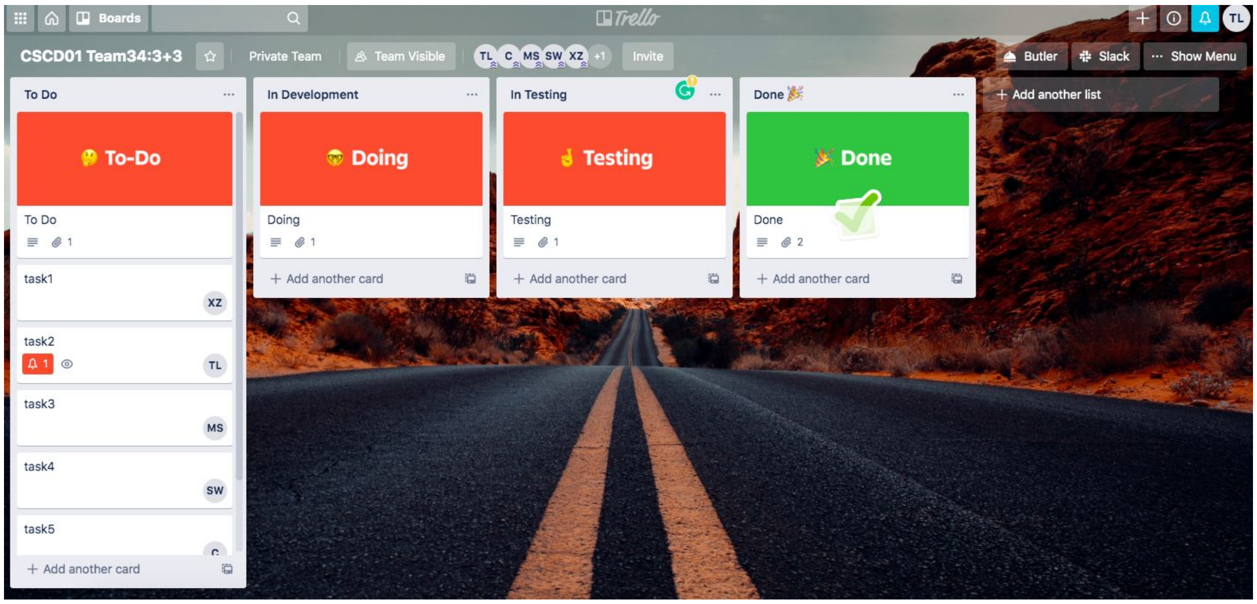
Choice of Software Development Process: Kanban

Stand-ups

- every two days instead of every day

We decided to modify the stand-ups to every two days instead of every day since all of the team members are taking other courses. The stand-ups will be a combination of both online and in-person. The in-person stand-ups will be on Mondays and Fridays at IC406 and the online stand-ups will be on Wednesdays using discord. Each stand-up will take about 18 minutes. Each team member will have about 3 minutes to talk about the tasks they have done and the resources they need as well as the problems they are facing and the tasks they will be working on in the next two days.

Kanban Board - For our project, we have decided to have 4 columns in our kanban board, including todo, in development, in testing and done. The reason why we have decided to have an extra column, the testing instead of the conventional kanban board setup is that we believe that having testing column individually helps us to visualize the progress of the tasks that we are working on at a time such as in development stage or in testing stage instead of having them all just in a in progress column. When a task is finished in development, it will be pulled to the in testing column and when its testing is done it will be pulled to the Done column which means the task has been finished.



Kanban Card - A Kanban card is the visual representation of a task. It will contain important information such as the summary of the task, assignee. For Kanban to work well, tasks have to be similar sized. Having one really large task kind of defeats the purpose of Kanban and WIP limits of managing work flow and being an incremental process.

Front of card: The front of the card will contain basic information like the title of the task, the summary of the task, priority, cycle time. Cards at the top of their respective column

will have higher priority. Cycle time being the time a task enters the “in progress” stage of development and testing and somebody starts working on it.
Back of card: The back of the card will have more detailed information about the task. Team members can attach comments, files, external links, any detailed valuable information.

Kanban Work-In-Progress Limits

The Work-In-Progress (WIP) limits the number of task items that a team is currently working on. Its purpose is to create a smooth workflow and allow team members to focus on current tasks. Having too high of WIP limits means the team will be working on too many multiple tasks and resulting in unproductive work not meeting deadlines. Too low of WIP limits can mean team members waiting for new tasks and production is too static. This is why Kanban WIP limits should be adjusted as needed. Our team will have WIP limits of 7 cards (6 group members plus one) and we may adjust it in the future.

Reasons for choosing Kanban

1. **Cons of the Waterfall process**
    a. It is linear and each stage has to be completed before the next stage can begin. It is not flexible for a team made up of 6 junior developers and each fully loaded with other course works;
    b. Gathering the requirements of the project is the most difficult part of the Waterfall process and it is hard to communicate with the repo owner or the issue reporter over the internet to get the requirements documented properly in the first stage.

2. **Cons of Scrum methodology**
    a. Scrum requires a fixed developed period - sprint and required release time. Kanban doesn’t require a fixed period, it’s continuous integration, team members can decide the release time depending on the priority of each task.
    b. Can’t create more tasks (does not exceed WIP)or modify current tasks when a sprint has started in Scrum, but Kanban allows tasks’ modification and creation at any time. That gives us more flexibility to manage each member’s workload.
    c. Tasks are limited by WIP in Kanban which Scrum doesn’t. That ensures the quality and completion of each task.

Kanban board helps the team to focus on the tasks while WIP helps to limit the number of tasks that the team should work on at a time, this helps us to ensure high quality of work which could ultimately save more time than, for example, using XP process and have a lot more tasks working at a time because tasks have low cost but eventually the team is not able to keep up the quality of our code and waste more time to just fix the issues occurred. After carefully considering the pros and cons of each developing process we are familiar with, we decided that the Kanban is the most suitable developing process for our team.